Chapter 8
Enhancing the Discovery of Internet of Things-Based Data Services in Real-time Linked Dataspaces

Wassim Derguech, Edward Curry, and Sami Bhiri

Keywords Sensor indexing · Service discovery · Formal concept analysis · Dataspaces · Intelligent systems · Internet of Things

8.1 Introduction

A dataspace is an emerging data management approach used to tackle heterogeneous data integration in an incremental manner. Data sources that are participants in a dataspace can be of various types such as online services, open datasets, sensors, and smart devices. Given the dynamicity of dataspaces and the diversity of their data sources and user requirements, finding appropriate sources of data can be challenging for users. Thus, it is important to describe and organise data sources in the dataspace efficiently. In this chapter, we present an approach for organising and indexing data services based on their semantic descriptions and using a feature-oriented model. We apply Formal Concept Analysis for organising and indexing the descriptions of sensor-based data services. We have experimented and validated the approach in a real-world smart environment which has been retrofitted with Internet of Things-based sensors observing energy, temperature, motion, and light.

The chapter is structured as follows. Section 8.2 explores the need for discovery of data services within dataspaces. Section 8.3 provides an overview of existing semantic approaches to service discovery. Section 8.4 sets out the theoretical foundations of Formal Concept Analysis and shows how it can be applied to indexing Internet of Things-based data services based on their capabilities using a concept lattice. It then shows how the concept lattice can be used to discover sensors and relationships between sensor properties. Section 8.5 reports on a smart environment intelligent system validation of the discovery approach. Finally, Sect. 8.6 draws conclusions and details of future work.

E. Curry, *Real-time Linked Dataspaces*,
https://doi.org/10.1007/978-3-030-29665-0_8

8.2 Discovery of Data Services in Real-time Linked Dataspaces

With the trend of Industry 4.0 [184], the advent of the Internet of Things (IoT) [185], and the decreasing costs and increasing capabilities of sensors and smart devices, modern businesses are integrating more and more real-time data into their business processes [186]. New challenges facing modern business processes include the dynamic and efficient discovery of resources such as data sources and services [185]. Indeed, many business processes rely on IoT sensor data to provide the necessary business intelligence and insight to support decision-making. With the rapid adoption of IoT devices and sensors, the number of available sources for real-time data has risen. A key challenge for intelligent systems within IoT-based smart environments is to discover and select appropriate sources of real-time data.

8.2.1 Real-time Linked Dataspaces

A data platform can provide a clear framework to support the sharing of data among a group of intelligent systems within a smart environment [1] (see Chap. 2). In this book, we advocate the use of the dataspace paradigm within the design of data platforms to enable data ecosystems for intelligent systems. A dataspace is an emerging approach to data management which recognises that in large-scale integration scenarios, involving thousands of data sources, it is difficult and expensive to obtain an upfront unifying schema across all sources [2]. Within dataspaces, datasets *co-exist* but are not necessarily fully integrated or homogeneous in their schematics and semantics. Instead, data is integrated on an *as-needed* basis with the labour-intensive aspects of data integration postponed until they are required. Dataspaces reduce the initial effort required to set up data integration by relying on automatic matching and mapping generation techniques. This results in a loosely integrated set of data sources. When tighter semantic integration is required, it can be achieved in an incremental *pay-as-you-go* fashion by detailed mappings among the required data sources.

We have created the Real-time Linked Dataspace (RLD) (see Chap. 4) as a data platform for intelligent systems within smart environments. The RLD combines the pay-as-you-go paradigm of dataspaces with linked data, knowledge graphs, and real-time stream and event processing capabilities to support a large-scale distributed heterogeneous collection of streams, events, and data sources [4].

8.2.2 Data Service Discovery

Within an RLD, creating explicit links between sensors-based data service descriptions helps to discover similar data services and consequently facilitate balancing

observations from one sensor to the other. A possible use case can be a smart building within an energy management application where a decision support model is used to control the supply and demand of energy. A principal source of information used for decision support models within smart buildings is sensors. An efficient decision support model in such a context requires that the sensor data provided is correct and timely. However, faults may occur at any time. For example, a sensor may become unresponsive or start to produce low-quality data. In these cases, the discovery support service should provide suggestions for alternative sources of data. This can be quickly implemented if sensors are adequately described and organised in a semantically enriched and linked manner.

In our past work, we have investigated the use of Formal Concept Analysis (FCA) [187] to better organise a repository of services' descriptions in order to make their discovery more efficient [188]. In this chapter, we revisit this work to contextualise it within the dataspace paradigm.

8.3 Semantic Approaches for Service Discovery

Several existing works use semantic technologies and related solutions for the indexing and discovery of services based on their semantic descriptions [189–192]. A key challenge with using these tools for service discovery in highly dynamic and interactive environments, such as IoT-based smart environments, is their cost in terms of computational resources. The high computational costs of some solutions [189, 190] are their dependency on reasoning within the discovery algorithms, a time-consuming task. Other solutions [192] try to resolve this issue by providing off-line indexing mechanisms for reducing the time complexity of discovery algorithms. However, given the dynamicity of smart environments, the diversity of their features and user requirements, reconstructing the entire indexing structure is challenging within medium and large-scale environments. Therefore, indexing/organising capabilities to enhance their discovery is required.

This section examines approaches that propose to use indexing structures for enhancing the discovery of IoT data services (e.g. a sensor) in a given repository (e.g. a dataspace). For each of the approaches analysed, we consider the following two key requirements:

- *Requirement 1—Ontology-based Discovery*: Searching resources should rely on concepts from domain ontologies used in their descriptions without relying on keyword extraction from textual descriptions [193]. Relying on extracting keywords from unstructured textual descriptions can lead to inconsistent results [194]. This requirement was elicited from the following works: [193, 195], and from Semantic Web Services Models: WSMO [196] and OWL-S [197].
- *Requirement 2—Time Performance*: Searching for a service or a business process can often rely upon reasoning, which makes the discovery very slow [191]. The second requirement is a low-latency response within large repositories of services

or business processes. This requirement was elicited from the following works: [190–192].

Using these requirements, we analyse related work in terms of their indexing *mechanism*, underlying *service description language*, and *limitations*.

8.3.1 Inheritance Between OWL-S Services

The discovery of semantic web services is challenging within large repositories due to costly reasoning operations. One solution to this problem is to introduce inheritance between OWL-S services [198]. Their specification denotes the possibility to define service profiles' hierarchies similar to object-oriented inheritances. Inheritance relationships between services are proposed to find service substitutes by exploring services that are higher in the hierarchy. Similar to object-oriented concepts, a sub-service may be used to substitute its super-service for automated, dynamic service discovery and composition [198]. Inheritance is also useful for creating new service profiles as a subclass of an existing profile. This makes the new service inherit the properties defined in the superclass profile. Other approaches [189] propose to capture such hierarchies in a visual editor for an OWL-S service description editor without discussing how these hierarchies can be created.

Introducing inheritance was a natural choice to enhance service discovery operation [198]. Services and relations are defined in the OWL language, and consequently, fulfils *Requirement 1—Ontology-based discovery*. However, little research has been carried out to determine inheritance between web services [199]; consequently, we cannot further comment on *Requirement 2—Time performance*.

8.3.2 Topic Extraction and Formal Concept Analysis

The work carried out by Aznag et al. [190] investigated the use of formal concept analysis as an indexing tool using topics extracted from service descriptions using SA-WSDL. Starting from a set of service descriptions, their algorithm converts them into a "service transaction matrix" that captures for each service the relevant textual concepts used in its description. This matrix is further refined with probabilistic clustering of the textual concept in order to extract a set of topics. The result of this analysis generates the correlated topic model that holds for each service and the topics it belongs to, with specific probabilities. In the work, formal concept analysis is used exclusively for clustering the extracted topics to make discovery easier when using a concept lattice. The use of formal concept analysis in this approach is due to its broad adoption as a well-established mathematical theory of concepts and concept hierarchies, which makes the service discovery much easier [190].

Topics used in the concept lattice are textual concepts that are extracted from the textual description of services, and consequently, this approach does not fulfil *Requirement 1—Ontology-based discovery*. Concerning *Requirement 2—Time performance*, the authors did not perform any evaluation of the time required to create the cluster of services. Nevertheless, they indicate that the query response time varies between 300 and 3000 ms with a test collection of 1088 services. Given that the discovery operation using formal concept analysis is a simple tree parsing operation, it has a linear complexity depending on the number of concepts in the created lattice (tree). Furthermore, in formal concept analysis, the creation of the concept lattice is the most expensive operation [200]. Thus, this can lead to the conclusion that the construction time of the entire cluster could be in the order of seconds.

8.3.3 Reasoning-Based Matching

Srinivasan et al. [192] looked into enhancing the indexing of a UDDI registry of services described in OWL-S during the service advertisement phase. Assuming that services are described using predefined ontologies, they use a matching degree between inputs and outputs of services. The matching uses concepts from the service description ontologies in order to identify a correct clustering of service descriptions into predefined ontological clusters similar to the North American Industry Classification System (NAICS) [201].

Requirement 1—Ontology-based discovery is fulfilled, as this approach relies exclusively on clusters that are constructed from hierarchical ontological concepts similar to NAICS [201]. However, the problem with this approach is that it relies heavily on reasoning and pre-computing information required for the search request, which is costly (*Requirement 2—Time performance*). The authors have confirmed this limitation through the performance evaluations that they carried out. The OWLS/UDDI approach takes more than 4000 ms for inserting 50 advertisements into the registry, which was 6–7 times slower than using a classical UDDI approach. However, the authors argue that this time is not very important as the advertisement operation can be done off-line and more time can be saved during the discovery phase without giving any quantifications.

8.3.4 Numerical Encoding of Ontological Concepts

Mokhtar et al. [191] optimise the indexing of service descriptions by avoiding semantic reasoning and by using a numeric coding scheme, a widely adopted method for enhancing the performance of ontology processing. They propose that a service registry can be clustered using a predefined ontology or taxonomy such as NAICS

[201] or UNSPSC [202], where an interval of numbers encodes each ontological concept. These intervals are defined using a linear inverse exponential function in a way where one interval can be contained in another one without overlap, creating a subscription relation. For example, to model sub-concept relations between Wi-Fi and Wireless, Wi-Fi can be coded by the interval [0, 0.1] and Wireless by the interval [0, 1] (i.e. [0, 0.1] is contained in [0, 1]). One can add to this example another concept, Bluetooth, that is, a sub-concept of Wireless by assigning the interval [0.2, 0.3] to Bluetooth (such that [0.2, 0.3] is contained in [0, 1], where [0, 0.1] and [0.2, 0.3] do not overlap).

Similar to [192], the authors of [191] rely exclusively on clusters that are constructed from hierarchical ontological concepts similar to NAICS [201]. Consequently, *Requirement 1—Ontology-based discovery* is fulfilled.

Concerning *Requirement 2—Time performance*, and compared to the performance of the work by Srinivasan et al. [192], Mokhtar et al. [191] achieve better results as the required time for encoding and advertisement does not exceed 450 ms for 50 service descriptions. This performance is achieved with the assumption that the used ontologies for service classification are encoded. Similarly, the reasoning operation is reduced to a comparison of codes/intervals. In this case, to infer that a concept c1 subsumes another concept c2, one needs to evaluate if their corresponding encoding interval of c1 is contained in the encoding interval of c2. This restricts the system to use classification ontologies that do not frequently evolve. Otherwise, service advertisements and requests need to check and update their encoding intervals periodically.

Similarly, Binder et al. [203] looked into encoding techniques to create hierarchies of service descriptions. Here the authors used the Generalised Search Tree algorithm [204] for organising service descriptions. This technique provides efficient search in the order of milliseconds over 10,000 entities. The main limitation of this approach is the complexity of maintaining this indexing structure: a new entry requires around 3 s for updating the tree [191].

8.3.5 Discussion

In summary, most of the analysed approaches rely on indexing service descriptions using existing taxonomies such as the North American Industry Classification System (NAICS) [201], UNSPSC [202], or the MIT Process Handbook [205]. This supports the idea of using ontologies as a common conceptualisation and shared understanding among service providers, registry hosts, and service requesters. However, the use of these ontologies makes the indexing heavily reliant on reasoning, a task that can be costly. The literature proposes multiple techniques that either used reasoning or proposed alternative solutions.

Table 8.1 Comparative analysis of capability indexing approaches

Approach	R1: Ontology-based discovery	R2: Time performance	Limitations
Inheritance between OWL-S services [189]	Not fulfilled	N/A	No clear methodology on how a hierarchy is created.
Topic extraction and formal concept analysis [190]	Fulfilled	Size: 1088 services, query response time between 300 ms and 3000 ms	Topic extraction and correlations are based on a probabilistic system. Solution needs further optimisations. FCA is exclusively used for topic clustering.
Reasoning-based matchmaking [192]	Fulfilled	Size: 50 services, index construction + advertisement time: ~4 s	Service advertisement operation is costly and heavily relying on reasoning.
Numerical encoding of ontological concepts and codes comparison [191]	Fulfilled	Size: 100 services, index construction + advertisement time: ~500 ms	Requires periodic updates of the codes of the clustering ontology. Slow registry maintenance for large repositories.

The analysis is summarised in Table 8.1. The key findings are:

- Indexing or clustering of service descriptions using ontologies is widely adopted [190–192].
- Maintainability of the indexing structure is critical to the applicability of the proposed approach [191, 192].
- The benefits of reusing existing techniques such as FCA for creating or maintaining the indexing structure is widely accepted [190].

8.4 Formal Concept Analysis for Organizing IoT Data Service Descriptions

In our past work, we have investigated the use of Formal Concept Analysis [187] to better organise a repository of service descriptions in order to make their discovery more efficient [188]. In this section, we revisit this work to contextualise it within the dataspace paradigm. We start by defining the theoretical foundations of FCA while applying it to IoT sensor data service descriptions.

FCA is a technique that has evolved from mathematical lattice theory and has been used for data analysis across several domains, such as organising web search

Table 8.2 Data table with binary attributes for sensor-based data services [188]

Objects	Active	Storage option	Digital display	Accessible
Sensor 1	X	X	X	X
Sensor 2	X		X	X
Sensor 3		X	X	X
Sensor 4		X	X	X
Sensor 5	X			

results into concepts based on common topics, gene expression data analysis, information retrieval, and understanding and analysis of source codes [206]. FCA helps in identifying meaningful relationships within a set of objects that share common attributes. It also provides a theoretical model to build from a *formal context* a partially ordered structure called a *concept lattice*.

8.4.1 Definition: Formal Context

A *formal context FC* is a triplet $<X,Y,R>$, where X and Y are non-empty sets and $R \subseteq X * Y$ is a binary relation between X and Y.

For a formal context FC, elements $x \in X$ are referred to as objects and elements $y \in Y$ are called attributes. $<x,y> \in R$ denotes that the object x has the attribute y.

In this work, the formal context is defined via the set of sensor data services as well as their respective descriptions. Table 8.2 will be used in this section as a running example that describes the relationships between the objects (i.e. sensors 1–5 represented by the table rows: X =*{Sensor 1, Sensor 2, Sensor 3, Sensor 4, Sensor 5}*) and their descriptions (i.e. attributes represented by the table columns: Y =*{Active, Storage Option, Digital Display, Accessible}*, in Table 8.2). This example considers the following four attributes:

- *Active:* Indicates if the sensor is in operation
- *Storage Option:* Indicates if the sensor can store data
- *Digital Display:* Indicates if the sensor is equipped with a digital display for displaying the data.
- *Accessible*: Indicates if the sensor is located in an accessible area

Another fundamental concept in FCA is the *Formal Concept*.

8.4.2 Definition: Formal Concept

A *formal concept* in $<X,Y,R>$ is a pair $<E,I>$ of $E \subseteq X$ (called *extent*) and $I \subseteq Y$ (called intent) such that $Att(E) = I$ and $Obj(I) = E$. $Att(E)$ is an operator that assigns subsets of X to subsets of Y, such that $Att(E)$ is the set of all attributes shared by all

objects from *E*. *Obj(I)* is an operator that assigns subsets of *Y* to subsets of *X*, such that *Obj(I)* is the set of all objects sharing all the attributes from *I*.

From this definition, one can conclude that a concept $C = <E,I>$ is created by getting objects from *E* sharing the same attributes from *I*. For example, the shaded rectangle in Table 8.2 represents a formal concept $<E\_1,I\_1> = <\{Sensor\ 1, Sensor\ 2, Sensor\ 3, Sensor\ 4\}, \{Digital\ Display, Accessible\}>$ because $Att(E\_1) = \{Digital\ Display, Accessible\}$ and $Obj(I\_1)=\{Sensor\ 1, Sensor\ 2, Sensor\ 3, Sensor\ 4\}$.

From a *formal context* $FC = <X,Y,I>$, one can deduce a set of formal concepts that can be ordered with respect to a *sub-concept ordering*.

8.4.3 Definition: Sub-concept Ordering

Having two formal concepts $<E\_1,I\_1>$ and $<E\_2,I\_2>$ from $FC = <X,Y,R>$, $<E\_1,I\_1> \leq <E\_2,I\_2>$ *iff* $E\_1 \subseteq E\_2$ (*iff* $I\_2 \subseteq I\_1$).

Let us consider the following formal concepts from the example in Table 8.2:

$<E\_1,I\_1> = <\{Sensor\ 1, Sensor\ 2, Sensor\ 3, Sensor\ 4\}, \{Digital\ Display, Accessible\}>$
$<E\_2,I\_2> = <\{Sensor\ 1, Sensor\ 2, Sensor\ 4\}, \{Digital\ Display, Accessible\}>$
$<E\_3,I\_3> = <\{Sensor\ 1, Sensor\ 2\}, \{Active, Digital\ Display, Accessible\}>$
$<E\_4,I\_4> = <\{Sensor\ 1, Sensor\ 2, Sensor\ 5\}, \{Active\}>$

Then

$<E\_3,I\_3> \leq <E\_1,I\_1>$, $<E\_3,I\_3> \leq <E\_2,I\_2>$, $<E\_3,I\_3> \leq <E\_4,I\_4>$ and $<E\_2,I\_2> \leq <E\_1,I\_1>$.

The set of ordered formal concepts derived from a formal context is called a *concept lattice*, which is another important notion in FCA. A concept lattice can be represented as a graph such as the one depicted in Fig. 8.1 (created using Conexp

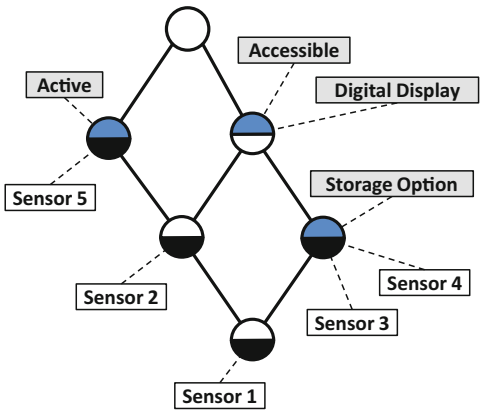

Fig. 8.1 Concept lattice of the example in Table 8.2 [188]

[207]). In this figure, the concept extent near the bottom of the lattice contains only *Sensor 1* since the corresponding intent is related to the most significant number of attributes. The top concept contains all the sensors, and its intent corresponds to no attribute. This makes the concept less attractive as it allows for all possible combinations of attributes.

In this example, we considered only binary attributes (i.e. either the object has or does not have that attribute). However, in real settings, when describing services, there are also multi-valued attributes that need to be transformed into a binary attribute through a scaling operation. For details about the process, we refer the reader to our publication, which details the scaling operation applied for this example [188].

This concept lattice is an indexing structure; it allows the organisation of sensor-based data services descriptions in a tree. This structure captures the explicit relations between formal concepts. This is useful to discover data services that share similar attributes. For example, if one of the sensors is not active anymore, it is possible to use one of the other sensors in its equivalence class or discover sensors that share its attributes. Furthermore, the presence of the explicit sub-concept relationship between concepts allows the discovery of additional knowledge among the objects' attributes that are analysed (i.e. sensor attributes). Indeed, as depicted in Fig. 8.1, one can discover implications such as: every sensor that has a "Storage Option" is also "Accessible" and has a "Digital Display". In other words: "Storage Option" implies "Accessible" and "Digital Display". Algorithms for knowledge discovery and implications have been presented in previous work [188].

It is important to note that the use of FCA permits the creation of a concept lattice uniformly. In other words, it always creates the same structure with the same input objects. This has the advantage of creating a deterministic discovery algorithm, as there is no need to use any heuristic for parsing this indexing structure. This chapter focuses mainly on the creation of the concept lattice and the study of its applicability for indexing a set of IoT sensor data service capabilities in a dataspace. In the next section, we use FCA in a real-world intelligent system setting for organising data service descriptions for a smart environment.

8.5 IoT-Enabled Smart Environment Use Case

This section illustrates a smart environment use case using a set of real-world IoT sensors deployed using the RLD where energy-related data is made available and interlinked to support intelligent system decision-making and ultimately improve energy consumption behaviour [100]. The RLD has been realised within a number of smart environments from smart homes to smart airports as detailed in Chap. 14. The data from the smart environments is provided by real-time data sources such as IoT sensors as well as relatively static background knowledge such as the building plan and occupancy.

In this section, we examine an intelligent system for energy management within a smart building pilot where there are various sources of power consumption, including heating, ventilation, and air conditioning (HVAC) systems, lights, and electronic devices. The building has been retrofitted with energy sensors to monitor the consumption of power. In total, there are over 50 fixed energy consumption sensors covering office space, a cafe, a data centre, kitchens, conference and meeting rooms, a computing museum along with over 20 mobile sensors for devices, light, temperature, and motion detection.

A building-specific deployment of the RLD has been presented in [62] with a sensor network-based situation awareness scenario presented in [208]. In total, this work used a total number of 78 sensors. The resulting IoT sensor data services are described via the following set of attributes:

- *Active:* This attribute reports whether the sensor is in operation.
- *Observed Phenomenon:* We have four observed phenomena, which are "energy and power consumption", "motion", "light", and "temperature". This attribute is a multi-valued attribute that needs to be scaled.
- *Protocol:* This attribute indicates the protocol used by the sensor. We have in our selection of sensors two possible protocols: UDP used by electricity and power consumption sensors and CoAP used by other sensors. This is a multi-valued attribute that has to be scaled.
- *Electricity Phases:* This attribute reports on the electricity phases used by the sensor; we have two options: three-phases and one-phase sensors. Again, this is a multi-valued attribute that has to be scaled.
- *Location:* Even though this attribute is not an intrinsic property of the sensor, we have used it because it is important information that is required for processing the data provided by the sensor. This is also a multi-valued attribute that enumerates the locations of the sensors, for example, 1st floor: west wing, ground floor: canteen, which needs to be scaled.

All the sensor data service capabilities were automatically generated from a tabular file containing the original descriptions that were manually checked. Manually checking RDF descriptions was possible as the number of sensors used was limited. We have not carried out any evaluation of the developed RDF parser, because it is custom made for the dataset and conceptual model. The correctness of the algorithm we applied for the RDF parser is out of the scope of this chapter; however, the data has been manually verified after parsing and scaling.

The resulting concept lattice from Conexp [207] is depicted in Fig. 8.2. The top concept in this lattice represents the set of all active sensors *<{Sensor 1, Sensor 2, ... Sensor 78}, {Active}>*. This formal concept contains in its extent all the sensors of the dataset because they are all active. One can see in this concept lattice several formal concepts that represent the set of motion sensors *<{Sensor 61,... Sensor 66}, {OP: Motion}>*, the formal concept for temperature sensors *<{Sensor 67,... Sensor 72}, {OP: Temperature}>* and the light sensors *<{Sensor 73,... Sensor 78}, {OP: Light}>*. These three formal concepts are sub-concepts of the concept *<{Sensor*

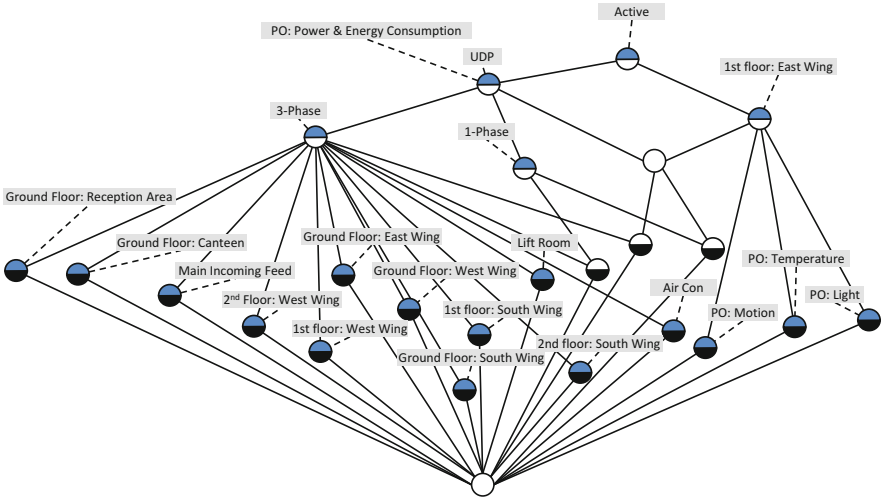

Fig. 8.2 Concept lattice of the smart office use case [188]

Table 8.3 Comparing time performances of indexing approaches

Indexing mechanism	Time performance
Inheritance between OWL-S services [189]	N/A
Topic extraction and FCA [190]	Size: 1088 services, query response time between 300 and 3000 ms
Reasoning-based matchmaking [192]	Size: 50 services, index construction + advertisement time: ∼4 s
Numerical encoding of ontological concepts and codes comparison [191]	Size: 100 services, index construction + advertisement time: ∼500 ms
Capabilities indexing using FCA [188]	Size: 1000 capabilities, index construction + parsing time: ≤25 ms

61, ... Sensor 78}, {1st Floor:East Wing}>. This helps to infer that all motion, temperature, and light sensors are in the same location (1st Floor: East Wing).

The focus of this work is on applying FCA in smart environments where the number of concepts does not exceed 1000. In such settings, the maximum construction time reaches only 25 ms [188]. Even though the main criticism towards using FCA, in this case, is the fact of reconstructing the concept lattice for any change in the environment (e.g. a new sensor, change in a sensor attribute), this remains acceptable with such low construction time.

Comparing the efficiency of the approach with respect to the approaches analysed in Sect. 8.3, we refer to Table 8.3. Each line of this table recalls the approach used, the indexing mechanisms, and the time performance as indicated by the authors in their papers. The table shows clearly that our approach outperforms the others because it does not use any reasoning for indexing the set of input capabilities.

8.6 Conclusions and Future Work

In this chapter, we discussed the use of Formal Concept Analysis (FCA) for indexing data sources, more specifically, IoT sensor data services in the context of dataspaces. Each of the data sources/services is described using a capability model capturing its functional and non-functional features. We use these descriptions for indexing the sensors using FCA, and the resulting indexing structure (called a concept lattice) which can support several use cases (e.g. the discovery of a replacement sensor).

In formal concept analysis, a formal context considers only single-valued attributes. In the case of multi-valued attributes, a scaling operation is required for transforming them into multiple single-valued attributes. In this chapter, we used only simple types of service capabilities that can be easily scaled from multi-valued to single-valued attributes. However, the model permits the modelling of complex values such as range and conditional values. Investigating scaling operations for covering these complex attribute types is required in order to consider them in the indexing approach using FCA.

Furthermore, a major concern in using FCA is its application for the analysis and indexing of large numbers of data services. FCA is known to be a memory and compute heavy technique [209]. In small-scale scenarios such as the use case in this chapter, the performance factor can be ignored as the computation time can be insignificant. However, in very large-scale deployments, this approach would ultimately fail because the time required to identify the concepts and create the lattice may be several hours. Incremental concept lattice creation can help in this direction. Indeed, FCA researchers [210] propose a concept lattice creation algorithm that has a quadratic worst-case time complexity in terms of the number of concepts, which could be leveraged in future work.